+++ date = '2025-05-16' draft = false title = 'Práctica 2' +++

Introducción

Este proyecto es un sistema de biblioteca digital que hice usando Python y Flask (un framework para crear páginas web). Básicamente, es una aplicación web donde puedes: agregar libros (normales o digitales), registrar miembros de la biblioteca, prestar y devolver libros, guardar toda la información en archivos para que no se pierda. Además, tiene un sistema de monitoreo de memoria para ver cuánta memoria está usando el programa.

API REST con Flask (biblioteca.py)

En este archivo es donde se manejan las acciones que se van a realizar desde el navegador. Se muestran los libros, se puede agregar un nuevo miembro, existe un guardado automático cada que se hace un cambio en los libros.

Agregar un libro.

```
@app.route('/books', methods=['POST'])
def add_book():
    data = request.json
    is_digital = data.get('is_digital', False)
    if is_digital:
        book = DigitalBook(
            int(data['id']), data['title'], data['author'],
data['publication_year'],
            data['genre'], data['quantity'], data['file_format']
        )
    else:
        book = Book(
            int(data['id']), data['title'], data['author'],
int(data['publication_year']),
            data['genre'], int(data['quantity'])
        )
    library.add_book(book)
    library.save_library_to_file("library.json")
    return jsonify(book.to_dict()), 201
```

Prestar un libro.

```
@app.route('/issue_book', methods=['POST'])
def issue_book():
    data = request.json
    book_id = int(data['book_id'])
    member_id = int(data['member_id'])
    library.issue_book(book_id, member_id)
```

```
    library.save_library_to_file("library.json")
    return jsonify({"message": "Libro prestado satisfactoriamente!"})
```

Modelo de Datos (biblioteca_web.py)

Define las clases principales del sistema (Book, DigitalBook, Member, Library, Genre) y sus métodos para gestionar libros, miembros y operaciones de la biblioteca.

Clase Book y DigitalBook: Representan libros físicos y digitales, respectivamente. Incluyen métodos para convertir objetos a diccionarios y viceversa (to_dict, from_dict).

Clase Member: Gestiona información de miembros y los libros que tienen prestados.

Clase Library: Contiene la lógica principal para agregar libros, prestar/devolver libros, y cargar/guardar datos en archivos JSON.

save_library_to_file y load_library_from_file permiten guardar y cargar el estado de la biblioteca.

Clase Book.

```
class Book:
    def __init__(self, book_id, title, author, publication_year, genre, quantity):
        self.id = book_id
        self.title = title
        self.author = author
        self.publication_year = publication_year
        self.genre = genre
        self.quantity = quantity
        memory_management.increment_heap_allocations(1)  # Monitor de memoria

    def to_dict(self):
        return {
            "id": self.id,
            "title": self.title,
            "author": self.author,
            "publication_year": self.publication_year,
            "genre": self.genre,
            "quantity": self.quantity
        }
```

Clase Library.

```
def issue_book(self, book_id, member_id):
    book = self.find_book_by_id(book_id)
    member = self.find_member_by_id(member_id)
    if book and member and book.quantity > 0:
        book.quantity -= 1
        member.issued_books.append(book_id)
```

```
        print("\nLibro prestado satisfactoriamente!\n")
```

Gestión de Memoria (memory_management.py).

Esta monitorea las asignaciones y liberaciones de memoria durante la ejecución del programa. La clase MemoryManagement lleva un registro de las asignaciones (heap_allocations) y liberaciones (heap_deallocations) de memoria.

El método display_memory_usage muestra el estado actual de la memoria, útil para depuración y optimización.

```
class MemoryManagement:
    def __init__(self):
        self.heap_allocations = 0
        self.heap_deallocations = 0

    def display_memory_usage(self):
        print(f"Heap allocations: {self.heap_allocations} bytes")
        print(f"Heap deallocations: {self.heap_deallocations} bytes")
```

Conclusión.

Este proyecto creo que es muy interesante ya que impementa Flask para la API web, las clases Book y Library son muy claras, las funciones so muy útiles y la declaración de las mismas, así como las variables me parece muy clara. Además de que este proyecto optimiza la forma en la que se usa la memoria, esto me parece muy importante y muy útil.